Webhooks

How to use the Aptic REST API

Date: 2021-12-27

aptic

Event log

Date	Change	Changed by
2021-12-27	Initial release	FE
Click to set date		
Click to set date		
Click to set date		

Aptic AB
Torsgatan 22, SE-113 22 Stockholm
Tel +46 (0) 10-101 20 00, hello@aptic.net

Content

Aptic AB
Torsgatan 22, SE-113 22 Stockholm
Tel +46 (0) 10-101 20 00, hello@aptic.net

Webhooks

1 Introduction

1.1 Aptic REST API

This document describes how to use one specific REST API provided for the Aptic services. This API is bound to specific versions of the services, so please refer to the latest version available at https://help.aptic.net/api.

1.1.1 Authentication

To be able to use the API, you need to authenticate first. There are two methods available, but your specific setup might be restricted to only one of them.

1.1.1.1 OAuth2 and client credential flow

This is the default authentication method and requires a client_id and a client_secret (a.k.a username and password).

By posting the client id and secret as basic authentication header or a web form to the authentication server, you will receive an access_token back, probably in the form of a JWT token.

Example of sending the oauth2 request as basic Auth header:

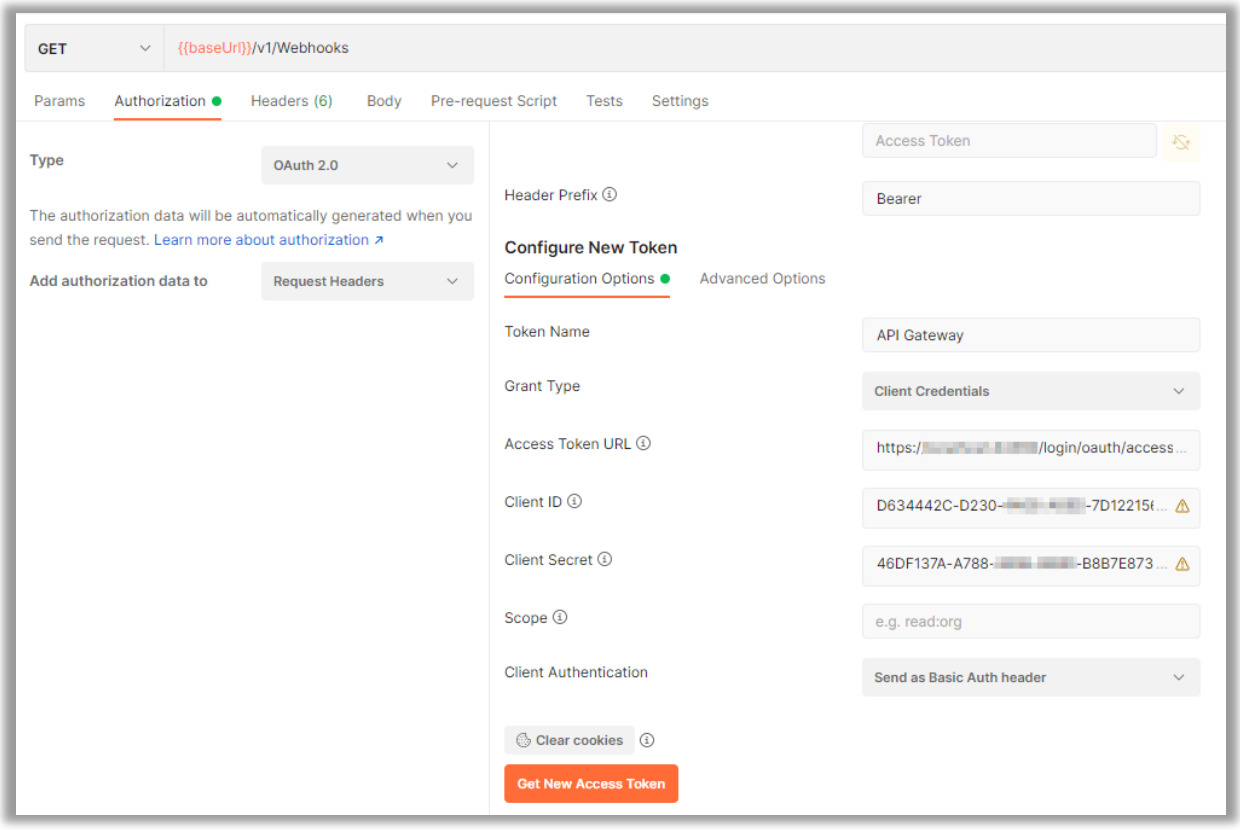
```
Content-Type: application/x-www-form-urlencoded
Authorization: Basic
RDYzNDQ0MkMtRNDY2DIzMC00LUE2OUxMjIxNTQtN0QZBMDJGOjQ2LUE3ODgREYxMzdBtNDY2RS044QjdFODcNjhE
LUIzQ0JEMQ==
Accept: */*
Cache-Control: no-cache
Host: localhost
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 29

grant_type: "client_credentials"
```

The response will be a json object containing the access_token and an optional refresh_token. The token_type should always be “bearer”.


```
{
"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjZXJ0c2VyaWFsbnVtYmVyIjoiRDYzNDQ
0MkMtRDIzMC00NDY2LUE2OUQtN0QxMjIxNTZBMDJGIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjA
wNS8wNS9pZGVudGl0eS9jbGFpbXMvc2lkIjoiMWU1M2EyN2MtYWI1Yi00NzkzLTg2YmEtYjIwNzcwMmI5ZjQ2Iiw
idW5pcXVlX25hbWUiOiJENjM0NDQyQy1EMjMwLTQ0NjYtOTY5RC03RDEyMjE1NkEwMkYiLCJuYmYiOjE2NDA2MDk
5NjgsImV4cCI6MTY0MDYxMzU2OCwiaWF0IjoxNjQwNjA5OTY4LCJpc3MiOiJBcHRpY0FQSUdhdGV3YXkiLCJhdWQ
iOiJhcGkifQ.0wpcypV6fFk9MsUtSv6YNYFxMDZmctSLd0xsaaPQaoU",
"token_type":"bearer",
"expires_in":3600,
"scope":""
}
```


The access_token should be provided in the authentication header in all subsequent calls to the API. Please note that the token is only valid for a limited time, one hour in this example. Then the token is not valid anymore and a new token must be acquired.

If you use a tool like Postman, the access token can be acquired with settings like this example:



In the API calls, you need to authenticate by providing the access token in the header like this:

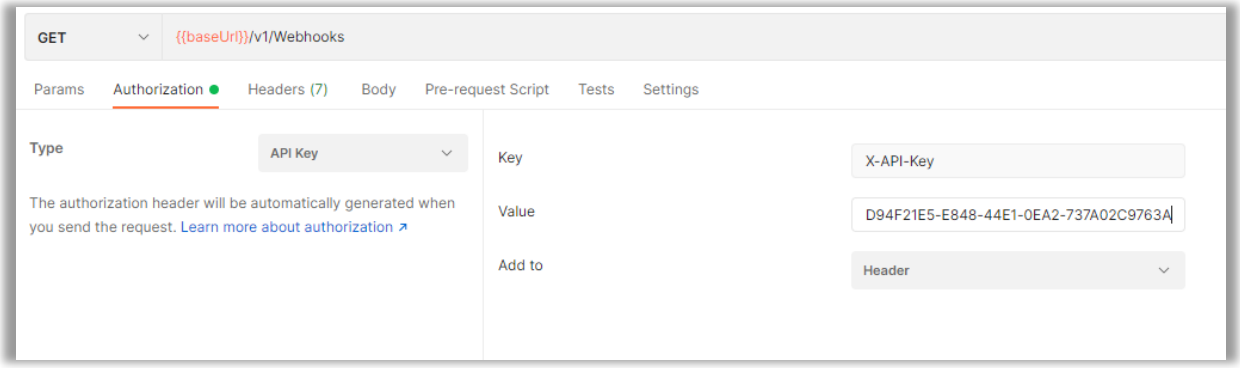
```
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjZXJ0c2VyaWFsbnVtYmVyIjoiRDYzNDQ0MkMtRDIzMC00NDY
2LUE2OUQtN0QxMjIxNTZBMDJGIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudGl
0eS9jbGFpbXMvc2lkIjoiMWU1M2EyN2MtYWI1YiO0NzkzLTg2YmEtYjIwNzcwMmI5ZjQ2IiwidW5pcXVlX25hbWU
iOiJENjM0NDQyQy1EMjMwLTQ0NjYtQTY5RCO3RDEyMjE1NkEwMkYiLCJuYmYiOjE2NDA2MDk5NjgsImV4cCI6MTY
0MDYxMzU2OCwiaWF0IjoxNjQwNjA5OTY4LCJpc3MiOiJBcHRpY0FQSUdhdGV3YXkiLCJhdWQiOiJhcGkifQ.0wpc
ypV6fFk9MsUtSv6YNYFxMDZmctSLd0xsaaPQaoU
Accept: */*
Host: localhost
Accept-Encoding: gzip, deflate, br
```

1.1.1.2 API Key

The second way of authentication is to use an API Key. The key is specified in the http header X-API-Key in all API calls, like this:

```
Accept: */*
Host: localhost
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
X-API-Key: D94F21E5-E848-44E1-0EA2-737A02C9763A
```

If you use Postman, you can specify the API key like this:



2 Using webhooks

2.1 The concept of webhooks

The idea of using webhooks, is to hook into the Aptic service and provide a callback method for the service to push messages on certain events. By doing this, it's not necessary to pull for the events, which both decreases the response time and lower to load on the back-end system.

The Aptic API provides a self-service API where you can add and remove your webhooks dynamically. Please be aware that the URLs provided needs to be available from the Aptic's back-end services.

The webhooks are added to the internal event queue in Aptic, and therefore have a fixed, limited, set of arguments. If this isn't enough, they can be extended by adding service definitions to the back-end.

2.2 Adding a webhook

To register a new webhook, you use the API: *PUT /v1/webhooks*. The arguments you need to specify are the public URL for the webhook and one or more event codes to bind the webhook to. There are several event codes and all of them have a different set of arguments returned to the webhook server.

In this example, we will add a webhook to the event *ledgerpayment* to react on payment of an invoice. Then callback URL is https://localhost:5510/webservice/webhook and I have also chosen to provide a secret, so I can verify that the caller is the expected one.

```
Content-Type: application/json
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjZXJ0c2VyaWFsbnVtYmVyIjoiRDYzNDQ0MkMtRDIzMC00NDY
2LUE2OUQtN0QxMjIxNTZBMDJGIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudGl
0eS9jbGFpbXMvc2lkIjoiZWE2OTNiZGYtMjFlZS00Yzg4LTgzOTktZjRmYjA0MTlhNDE1IiwidW5pcXVlX25hbWU
iOiJENjM0NDQyQy1EMjMwLTQ0NjYtQTY5RC03RDEyMjE1NkEwMkYiLCJuYmYiOjE2NDA2MTUxNDksImV4cCI6MTY
0MDYxODc0OSwiaWF0IjoxNjQwNjE1MTQ5LCJpc3MiOiJBcHRpY0FQSUdhdGV3YXkiLCJhdWQiOiJhcGkifQ.TKDQ
f5EjlNpuIBVUnE8srmzGTiRCQcLzKcAvLzt2KOc
Accept: */*
Host: localhost
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 137

{ "events": [ "ledgerpayment" ], "url": "https://localhost:5510/webservice/webhook",
"secret": "mySecret", "method": "POST" }
```

On success, you will receive HTTP response 201 (created) and all the events you have registered to that URL:

```
{
 "events":["ledgerpayment"],
 "url":"https://localhost:5510/webservice/webhook",
 "method":"POST"
}
```

If you want to check all your registered webhooks, you can use the *GET /v1/webhooks* API.

```
{
 "count":1,
 "webHooks":[
  {
    "events":["ledgerpayment"],
    "url":"https://localhost:5510/webservice/webhook",
    "method":"POST"
  }
 ]
}
```

2.3 Receiving a webhook

After the webhook is registered, any payment to an invoice will trigger a request to the registered web server. The body will contain the internal identities of the payment and the provided secret.

```
{
  "paymentid":"18142",
  "ledgeraccountid":"111927",
  "revision":"1",
  "debtcreditorid":"1",
  "partytypeid":"1",
  "secret":"mySecret"
}
```

A similar response for the event code *statuschange* would look like this:

```
{
  "ledgeraccountid":"111927",
  "status":"PAID",
  "secret":"mySecret"
}
```

2.4 Customizing a webhook

The internal identities provided in the webhooks might be of no use for an external system. To adjust the request body to the webhook, you can add a service definition to compose your customized response.

In the Aptic service back-end, you create a service definition of the category *webhook* and the event type as name. This is an example where we choose to provide the invoice GUID, invoice number and paid amount in the webhook.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<aptic_service_definition
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://schema.aptic.net/aptic-service-definition.xsd"
>
  <category>webhook</category>
  <name>ledgerpayment</name>

  <steps>
    <query>
      <![CDATA[
        select
          la.uniqueid as invoiceguid, dh.debtref as invoicenumber,
          sum(-t.transamount) as paidamount, ? as [secret]
        from dat_ledgeraccount la
        inner join dat_debthead dh on dh.debtheadid = la.maindebtheadid
        inner join dat_trans t on t.accountid = la.accountid and t.paymentid = ?
        where la.ledgeraccountid = ?
        group by la.uniqueid, dh.debtref
      ]]>
      <argument source="$secret" type="string"/>
      <argument source="$data.data.paymentid" type="int"/>
      <argument source="$data.data.ledgeraccountid" type="int"/>
    </query>
  </steps>
</aptic_service_definition>
```

After adding the service definition above, the body in the webhook looks like this:

```
{
  "invoiceguid":"80342534-2508-4CAB-A090-C25BF9DDF923",
  "invoicenumber":"765",
  "paidamount":123.00,
  "secret":"mySecret"
}
```

2.5 Removing a webhook

When you want to remove a webhook, you use the *DELETE /v1/webhook* API.

You specify the webhook URL you want to remove in the request body. You can also specify an array of event codes if you only want to remove some events.

```
{
    "url": "https://localhost:5510/webservice/webhook",
    "events": [
        "ledgerpayment"
    ]
}
```